1 Approach taken for the calculation in the program

In my Python program that calculates the required values, I have divided it into 2 main parts, one of which consists of several function modules and the other consists of calculations of the jet downstream for both the simple and non-simple regions.

1.1 Function Modules

I started the task by creating as many function modules as possible without the knowledge of how frequently they would be used in the actual calculations. In the later part of the programming, a few functions that are similar to the existing ones were added due to impracticality to edit the large portion of already written lines for the sake of slight increase in neatness and decrease in number of lines.

Brief explanations have been given for each of the function modules as there were no need to explicitly mention the inputs and outputs which were very simple to grasp from one look.

```python
""" Function modules """

gamma_ratio = np.sqrt( (gamma + 1) / (gamma - 1) )

def Mach_Prandtl(M):
    '''
    Converts Mach number to nu (Prandtl-Meyer angle)
    '''
    nu = gamma_ratio * np.arctan( 1 / gamma_ratio * np.sqrt(M*M - 1) ) - np.arctan(
    ↪  np.sqrt(M*M - 1) )

    return nu

def Prandtl_Mach(nu):
    '''
    Converts nu (Prandtl-Meyer angle) to Mach number
    '''
    func = lambda x: gamma_ratio * np.arctan( 1 / gamma_ratio * np.sqrt(x*x - 1) ) -
    ↪  np.arctan( np.sqrt(x*x - 1) ) - nu
    Mach = scipy.optimize.fsolve(func, 2.)[0]

    return Mach

def Mach_mu(M):
    '''
    Converts Mach number to mu (angle between flow and characteristic)
    '''
    if M == 0:
        mu = 0 # To accomodate for the slope of centerline
    else:
        mu = np.arcsin(1 / M)

    return mu

def char_angle(mu, phi, slope="plus"):
    '''
    Calculates the angle characteristic line makes with horizontal
    slope = "plus" or "minus"
    '''
    if slope == "plus":
```

```python
        angle = mu + phi
    elif slope == "minus":
        angle = phi - mu

    return angle

def isentropic_M(M_1, P_1, P_2):
    '''
    Calculates the Mach number of region 2 which is isentropically related to region 1
    '''
    M_2 = np.sqrt( 2 / (gamma - 1) * ( (P_1 / P_2) ** ( (gamma - 1) / gamma) * ( 1 +
    ↪  (gamma - 1) / 2 * M_1 * M_1 ) - 1) )

    return M_2

def characteristic(Prandtl_1, phi_1, Prandtl_2=None, phi_2=None, slope="plus"):
    '''
    Calculates properties of the characteristic line
    '''
    if slope == "plus":
        if Prandtl_2 != None:
            value = Prandtl_2 - Prandtl_1 + phi_1 # value is phi
        elif phi_2 != None:
            value = Prandtl_1 - phi_1 + phi_2     # value is Prandtl
    elif slope == "minus":
        if Prandtl_2 != None:
            value = Prandtl_1 + phi_1 - Prandtl_2 # value is phi
        elif phi_2 != None:
            value = Prandtl_1 + phi_1 - phi_2     # value is Prandtl

    return value

def intersect_para(Prandtl_1, phi_1, Prandtl_2, phi_2):
    '''
    Calculates Prandtl-Meyer angle and phi at interaction point of 2 characteristics
    1 is plus characteristic
    2 is minus characteristic
    '''
    Prandtl = 0.5 * (Prandtl_1 + Prandtl_2) + 0.5 * (phi_2 - phi_1)
    phi     = 0.5 * (phi_1 + phi_2) + 0.5 * (Prandtl_2 - Prandtl_1)

    return Prandtl, phi

def intersect_loc(M_1, phi_1, coord_1, M_2, phi_2, coord_2):
    '''
    Calculates the location of the intersection between 2 characteristics
    '''
    mu_1 = Mach_mu(M_1)
    mu_2 = Mach_mu(M_2)

    slope_1 = np.tan( char_angle(mu_1, phi_1, "plus") )
    slope_2 = np.tan( char_angle(mu_2, phi_2, "minus") )

    x_1, y_1 = coord_1
    x_2, y_2 = coord_2

    x_loc = ( slope_1 * x_1 - y_1 - slope_2 * x_2 + y_2 ) / ( slope_1 - slope_2 )
    y_loc = slope_1  * (x_loc - x_1) + y_1

    return x_loc, y_loc

```

```python
def intersect_centerline(Prandtl, phi, coord):
    '''
    Calculates the properties of intersection between expansion wave and centerline
    '''
    Prandtl_c = Prandtl + phi
    phi_c = 0

    mu = Mach_mu(Prandtl_Mach(Prandtl))
    slope = np.tan( char_angle(mu, phi, "minus") )
    slope_c = 0
    x, y = coord
    x_c, y_c = 0, H/2

    x_loc = ( slope * x - y - slope_c * x_c + y_c ) / ( slope - slope_c )
    y_loc = slope  * (x_loc - x) + y

    return Prandtl_c, phi_c, x_loc, y_loc


def intersect_boundary(Prandtl_bound, M_bound, coord_bound, Prandtl, phi, coord, order="not0"):
    '''
    Calculates properties at the jet boundary line
    '''
    phi_bound = Prandtl_bound - Prandtl + phi
    mu_bound = Mach_mu(M_bound)
    M = Prandtl_Mach(Prandtl)
    mu = Mach_mu(M)

    if order == "0":
        slope_bound = np.tan(region_ABC()[0][-1])
    else:
        slope_bound = np.tan(phi_bound)
    slope_wave  = np.tan( char_angle(mu, phi, "plus") )

    x_bound, y_bound = coord_bound
    x_wave, y_wave   = coord

    x_loc = ( slope_bound * x_bound - y_bound - slope_wave * x_wave + y_wave ) / ( slope_bound - slope_wave )
    y_loc = slope_bound  * (x_loc - x_bound) + y_bound

    return phi_bound, x_loc, y_loc

def intersect_stream(stream_coord, stream_slope, char_coord, char_slope):
    '''
    Calculates intersection between the streamline and characteristics
    '''
    x_1, y_1 = stream_coord
    x_2, y_2 = char_coord
    x_loc = ( stream_slope * x_1 - y_1 - char_slope * x_2 + y_2 ) / ( stream_slope - char_slope )
    y_loc = stream_slope  * (x_loc - x_1) + y_1

    return x_loc, y_loc

def linear_func(slope, coord, input_x):
    '''
    Calculates corresponding y-coordinate pair of x-coordinate on a given function defined by a slope and a pair of coordinates
    '''
```

```
    xi, yi = coord
    f = slope * (input_x - xi) + yi

    return f
```

1.2 Main calculations of jet downstream

When I realised that a large number of lines of code could have been cut down from implementing a flexible function for simple and non-simple regions, it was too late thus I have written a function for each region and they are namely:

- ABC: Simple region 1 - Global region 0
- BCE: Non-simple region 1 - Global region 1
- CDEF: Simple region 2 - Global region 2
- DFG: Non-simple region - Global region 3
- FGH: Simple region 3 - Global region 4
- HIJK: Non-simple region 3 - Global region 5
  - Points I, J and K are not specified in the figure included in the task document. They are the corner points that forms the third non-simple region together with point H.

Little to no calculations were done for the simple regions. However, the parameters in these regions were set up to aid the debugging and also to maintain consistency in each region affecting the very next subsequent region. Furthermore, as for the global region 3, calculations regarding the jet boundary line were also included which gives gradient of the line as an output (although it is named as phi_jet).

```
""" Calculation of Regions """

def region_ABC():
    '''
    Caclulates properties of initial expansion wave in the simple region
    '''
    phi_0, Prandtl_0, M_0 = np.zeros((N+1)), np.zeros((N+1)), np.zeros((N+1))
    M_0[0]                = Me
    Prandtl_0[0]          = Mach_Prandtl(M_0[0])
    Prandtl_0[-1]         = Mach_Prandtl( isentropic_M(M_0[0], Pe, Pa) ) # Pressure
    ↪ values are used here
    M_0[-1]               = Prandtl_Mach(Prandtl_0[-1])
    phi_0[-1]             = characteristic(Prandtl_0[0], phi_0[0], Prandtl_0[-1], None,
    ↪ "plus")
    Delta_phi_0           = (phi_0[-1] - phi_0[0]) / (N - 1)

    for i in range(1, N):
        phi_0[i] = phi_0[i-1] + Delta_phi_0
        Prandtl_0[i] = characteristic(Prandtl_0[i-1], phi_0[i-1], None, phi_0[i], "plus")
        M_0[i] = Prandtl_Mach(Prandtl_0[i])

    return phi_0, Prandtl_0, M_0


def region_BCE():
    '''
    Calculates properties of the wave during reflection in the non-simple region
    '''
    phi_0, Prandtl_0, M_0 = region_ABC()
    phi_1, Prandtl_1, M_1 = np.zeros((N,N)), np.zeros((N,N)), np.zeros((N,N))
    x_loc, y_loc          = np.zeros((N,N)), np.zeros((N,N))

    for j in range(N):
```

```python
        if j == 0:
            Prandtl_1[j,j], phi_1[j,j], x_loc[j,j], y_loc[j,j] = intersect_centerline(Prandtl_0[j], phi_0[j], coord_e)
            M_1[j,j] = Prandtl_Mach(Prandtl_1[j,j])
        else:
            Prandtl_1[j,j], phi_1[j,j], x_loc[j,j], y_loc[j,j] = intersect_centerline(Prandtl_1[j-1,j], phi_1[j-1,j], (x_loc[j-1,j], y_loc[j-1,j]))
            M_1[j,j] = Prandtl_Mach(Prandtl_1[j,j])

        for i in range(j+1,N):
            if j == 0:
                Prandtl_1[j,i], phi_1[j,i] = intersect_para(Prandtl_1[j,i-1], phi_1[j,i-1], Prandtl_0[i], phi_0[i])
                M_1[j,i] = Prandtl_Mach(Prandtl_1[j,i])
                x_loc[j,i], y_loc[j,i]  = intersect_loc(M_1[j,i-1], phi_1[j,i-1], (x_loc[j,i-1], y_loc[j,i-1]), M_0[i], phi_0[i], coord_e)
            else:
                Prandtl_1[j,i], phi_1[j,i] = intersect_para(Prandtl_1[j,i-1], phi_1[j,i-1], Prandtl_1[j-1,i], phi_1[j-1,i])
                M_1[j,i] = Prandtl_Mach(Prandtl_1[j,i])
                x_loc[j,i], y_loc[j,i]  = intersect_loc(M_1[j,i-1], phi_1[j,i-1], (x_loc[j,i-1], y_loc[j,i-1]), M_1[j-1,i], phi_1[j-1,i], (x_loc[j-1,i], y_loc[j-1,i]))

    return phi_1, Prandtl_1, M_1, x_loc, y_loc


def region_CDEF():
    '''
    Sets up the parameters for characteristics in the simple region
    '''
    phi_2, Prandtl_2, M_2, x_2, y_2 = np.zeros((N)), np.zeros((N)), np.zeros((N)), np.zeros((N)), np.zeros((N))
    phi_1, Prandtl_1, M_1, x_1, y_1 = region_BCE()

    for i in range(N):
        phi_2[i]      = phi_1[i,-1]
        Prandtl_2[i] = Prandtl_1[i,-1]
        M_2[i]        = M_1[i,-1]
        x_2[i]        = x_1[i,-1]
        y_2[i]        = y_1[i,-1]

    return phi_2, Prandtl_2, M_2, x_2, y_2

def region_DFG():
    '''
    Calculates properties of the wave during reflection in the non-simple region including the shape of the jet boundary line
    '''
    phi_0, Prandtl_0, M_0               = region_ABC() # information about initial jet boundary line
    phi_2, Prandtl_2, M_2, x_2, y_2     = region_CDEF()
    phi_3, Prandtl_3, M_3, x_loc, y_loc = np.zeros((N,N)), np.zeros((N,N)), np.zeros((N,N)), np.zeros((N,N)), np.zeros((N,N))

    # Jet boundary line parameters
    phi_jet, angle_jet = np.zeros((N+1)), np.zeros((N+1))
    Prandtl_jet        = Prandtl_0[-1] # Constant along boundary line
    M_jet              = M_0[-1] # Constant along boundary line
    mu_jet             = Mach_mu(M_jet) # Constant along boundary line
```

```python
    phi_jet[0]          = Prandtl_jet - Prandtl_2[0] + phi_2[0]

    for j in range(N):
        if j == 0:
            phi_3[j,j], x_loc[j,j], y_loc[j,j] = intersect_boundary(Prandtl_jet, M_jet, coord_e, Prandtl_2[j], phi_2[j], (x_2[j], y_2[j]), "0")
            Prandtl_3[j,j] = Prandtl_jet
            M_3[j,j]       = M_jet
        else:
            phi_3[j,j], x_loc[j,j], y_loc[j,j] = intersect_boundary(Prandtl_jet, M_jet, (x_loc[j-1,j-1], y_loc[j-1,j-1]), Prandtl_3[j-1,j], phi_3[j-1,j], (x_loc[j-1,j], y_loc[j-1,j]))
            Prandtl_3[j,j] = Prandtl_jet
            M_3[j,j]       = M_jet
            phi_jet[j+1]        = np.tan(phi_3[j,j])

        for i in range(j+1,N):
            if j == 0:
                Prandtl_3[j,i], phi_3[j,i] = intersect_para(Prandtl_2[i], phi_2[i], Prandtl_3[j,i-1], phi_3[j,i-1])
                M_3[j,i] = Prandtl_Mach(Prandtl_3[j,i])
                x_loc[j,i], y_loc[j,i]  = intersect_loc( M_2[i], phi_2[i], (x_2[i], y_2[i]), M_3[j,i-1], phi_3[j,i-1], (x_loc[j,i-1], y_loc[j,i-1]) )
            else:
                Prandtl_3[j,i], phi_3[j,i] = intersect_para(Prandtl_3[j-1,i], phi_3[j-1,i], Prandtl_3[j,i-1], phi_3[j,i-1])
                M_3[j,i] = Prandtl_Mach(Prandtl_3[j,i])
                x_loc[j,i], y_loc[j,i]  = intersect_loc(M_3[j-1,i], phi_3[j-1,i], (x_loc[j-1,i], y_loc[j-1,i]), M_3[j,i-1], phi_3[j,i-1], (x_loc[j,i-1], y_loc[j,i-1]))

    return phi_3, Prandtl_3, M_3, x_loc, y_loc, phi_jet

def region_FGHI():
    '''
    Sets up the parameters for characteristics in the simple region
    '''
    phi_3, Prandtl_3, M_3, x_3, y_3 = region_DFG()[:-1]
    phi_4, Prandtl_4, M_4, x_4, y_4 = np.zeros((N)), np.zeros((N)), np.zeros((N)), np.zeros((N)), np.zeros((N))

    for i in range(N):
        phi_4[i]     = phi_3[i,-1]
        Prandtl_4[i] = Prandtl_3[i,-1]
        M_4[i]       = M_3[i,-1]
        x_4[i]       = x_3[i,-1]
        y_4[i]       = y_3[i,-1]

    return phi_4, Prandtl_4, M_4, x_4, y_4

def region_HIJK():
    '''
    Calculates properties of the wave during reflection in the non-simple region
    '''
    phi_4, Prandtl_4, M_4, x_4, y_4 = region_FGHI()
    phi_5, Prandtl_5, M_5, x_loc, y_loc = np.zeros((N,N)), np.zeros((N,N)), np.zeros((N,N)), np.zeros((N,N)), np.zeros((N,N))

    for j in range(N):
        if j == 0:
```

```python
                Prandtl_5[j,j], phi_5[j,j], x_loc[j,j], y_loc[j,j] = intersect_centerline(Prandtl_4[j], phi_4[j], (x_4[0], y_4[0]))
                M_5[j,j] = Prandtl_Mach(Prandtl_5[j,j])
            else:
                Prandtl_5[j,j], phi_5[j,j], x_loc[j,j], y_loc[j,j] = intersect_centerline(Prandtl_5[j-1,j], phi_5[j-1,j], (x_loc[j-1,j], y_loc[j-1,j]))
                M_5[j,j] = Prandtl_Mach(Prandtl_5[j,j])

            for i in range(j+1,N):
                if j == 0:
                    Prandtl_5[j,i], phi_5[j,i] = intersect_para(Prandtl_5[j,i-1], phi_5[j,i-1], Prandtl_4[i], phi_4[i])
                    M_5[j,i] = Prandtl_Mach(Prandtl_5[j,i])
                    x_loc[j,i], y_loc[j,i]  = intersect_loc(M_5[j,i-1], phi_5[j,i-1], (x_loc[j,i-1], y_loc[j,i-1]), M_4[i], phi_4[i], (x_4[i], y_4[i]))
                else:
                    Prandtl_5[j,i], phi_5[j,i] = intersect_para(Prandtl_5[j,i-1], phi_5[j,i-1], Prandtl_5[j-1,i], phi_5[j-1,i])
                    M_5[j,i] = Prandtl_Mach(Prandtl_5[j,i])
                    x_loc[j,i], y_loc[j,i]  = intersect_loc(M_5[j,i-1], phi_5[j,i-1], (x_loc[j,i-1], y_loc[j,i-1]), M_5[j-1,i], phi_5[j-1,i], (x_loc[j-1,i], y_loc[j-1,i]))

        return phi_5, Prandtl_5, M_5, x_loc, y_loc


""" Main """

phi_0, Prandtl_0, M_0                               = region_ABC()  # 1D matrices
phi_1, Prandtl_1, M_1, x_1, y_1                     = region_BCE()  # upper triangle 2D matrices
phi_2, Prandtl_2, M_2, x_2, y_2                     = region_CDEF() # 1D matrices
phi_3, Prandtl_3, M_3, x_3, y_3, jetbound_angle = region_DFG()  # upper triangle 2D matrices
phi_4, Prandtl_4, M_4, x_4, y_4                     = region_FGHI() # 1D matrices
phi_5, Prandtl_5, M_5, x_5, y_5                     = region_HIJK() # upper triangle 2D matrices
```

2 Visualisation of the characteristic and the jet boundary

Using the Python scripts as shown in section 1 and another set of scripts to reorder the values in the matrices to allow for plotting, the Figure 1 was produced.

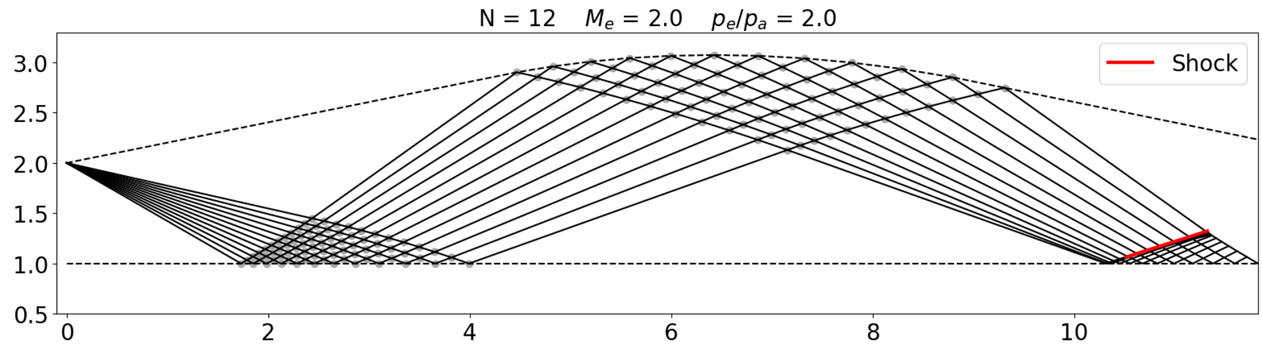

Figure 1: Downstream of jet exit with 12 characteristic lines, $M_e = 2.0$, $p_e/p_a = 2.0$

H of 2 and small enough number of characteristic lines, N, of 12 was chosen for the plot in order to allow for visualisation of the intersections of characteristics, shown by the translucent grey circles. The red line indicates the location of the shock for this setting and the 2 black dotted lines represents the boundary, the one above for the jet boundary and the one below for the centerline.

As for the jet boundary, it maintains a constant slope that is directly related to the constant ϕ value of the uniform region directly beneath it until it intersects with N characteristic lines in the second non-simple region. Using the fact that the Prandtl-Meyer angle of the jet boundary is constant due to the constant Mach which is due to the constant pressure p_a along the jet boundary, the interactions between the jet boundary and each characteristic were able to be calculated. The value of phi that results from the each interaction gives arise to the new slope value in the downstream.

It can be observed from figure 1 that every interaction reduces the slope of the jet boundary, eventually resulting in a negative jet boundary slope and it maintains the slope after the last interaction.

For the characteristics, they all start from a single point at $(x, y) = (0, H)$ as negative slopes and are diverging. After reflecting off the centerline, they are still diverging but starts converging after the interaction with the jet boundary. This is contributed by the prior intersections between characteristics where the subsequent characteristics have reductions in slopes.

3 Mach distribution

For the plot of mach distribution, a much higher number of characteristics of 30 was used to show the smooth transition along the jet downstream and this is shown in the figure 2.

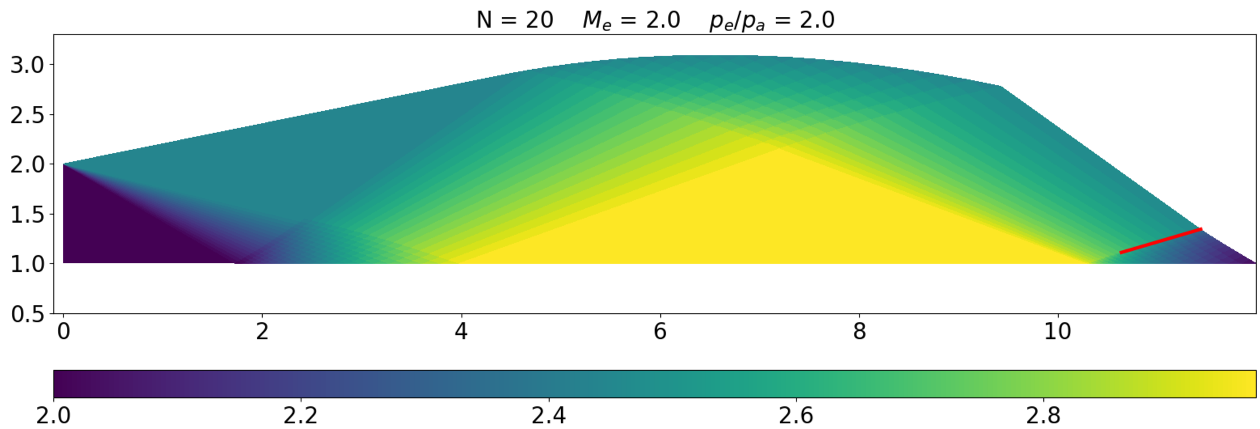

Figure 2: Mach distribution with 25 characteristic lines, $M_e = 2.0$, $p_e/p_a = 2.0$

The plot has been coloured based on the magnitude of the Mach number in the jet downstream and the colourbar below the figure describes the values of Mach number that corresponds to the different colour gradient. Again, the red line indicates the location of the shock.

As expected, the Mach number along the boundary seems to be constant. Furthermore, every expansion wave results in a higher Mach number. It can also be observed that the characteristics downstream of the yellow uniform region are compression waves leading to the shock. These compression waves result from the varying parameters of the jet boundary with which the expansion waves interacts with and reflects off from.

4 Streamlines

Two streamlines that start at different height are shown in this section with their respective changes in pressure throughout the jet downstream.

4.1 Starting on the centerline of the jet exit, $y = H/2$

For $H = 2$, the centerline of the jet exit is located at $y = 1.0$ as shown in the figure 3. The streamline starting at this point, $(x, y) = (0, 1.0)$ is described by the orange line.

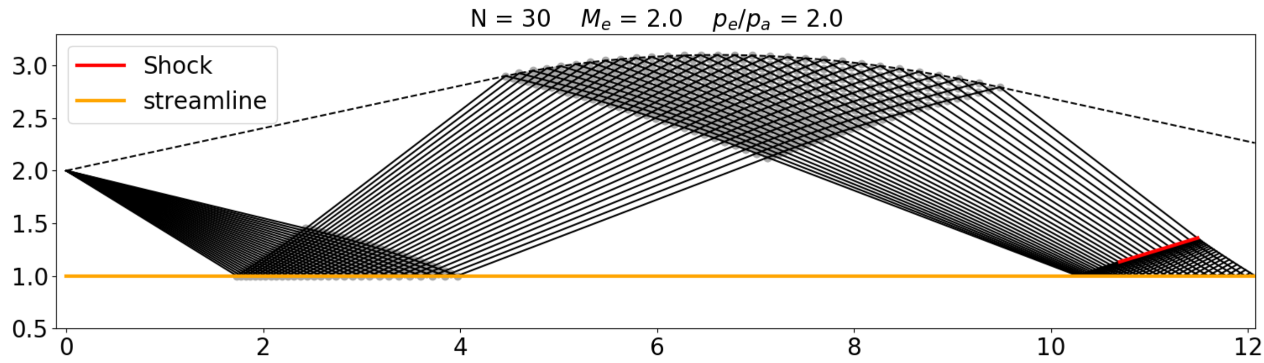

Figure 3: Streamline starting at $y = H/2$, $M_e = 2.0$, $p_e/p_a = 2.0$

It can be observed that the streamline stays on the centerline throughout the jet downstream. The pressure ratio, p/p_T, where p_T is the total pressure, of the streamline has been plotted in the figure 4.

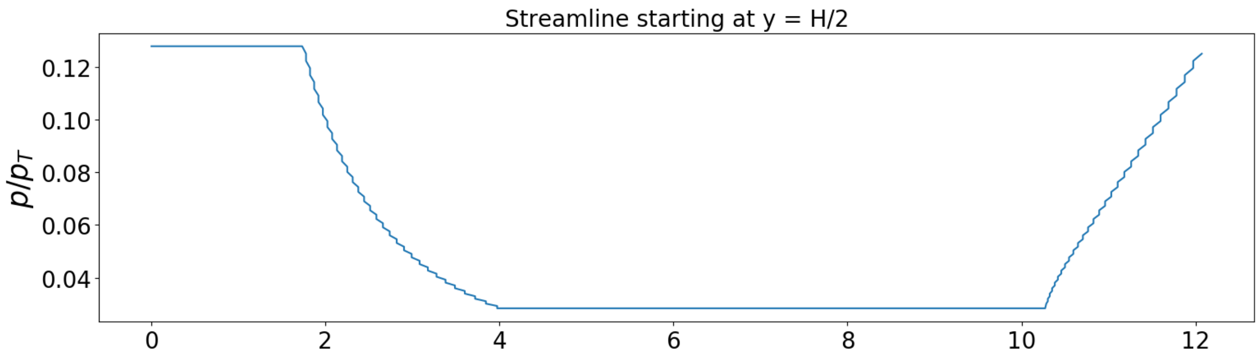

Figure 4: Pressure plot along streamline starting at $y = H/2$, $M_e = 2.0$, $p_e/p_a = 2.0$

The pressure along the streamline stays constant initially throughout the uniform region until it reaches the non-simple region wherein the Mach number increased due to expansion. The pressure of the streamline in this region exponentially decreased and then stays constant again through the next uniform region. It can be observed that the exponential decrease is not depicted as a smooth line but with small zig-zag behaviour due to the various intersections of the characteristics in this non-simple region. Afterwards, the pressure spikes up in the last non-simple region where the flow is compressed and it is also where the shock is supposed to appear indicated by the red line. Thus, the pressure plot describes the streamline in a way that I have expected it to be.

4.2 Starting at $y = 3H/4$

The streamline starting at $(x, y) = (0, 3H/4) = (0, 1.5)$ is analysed in this subsection using the figure 5.

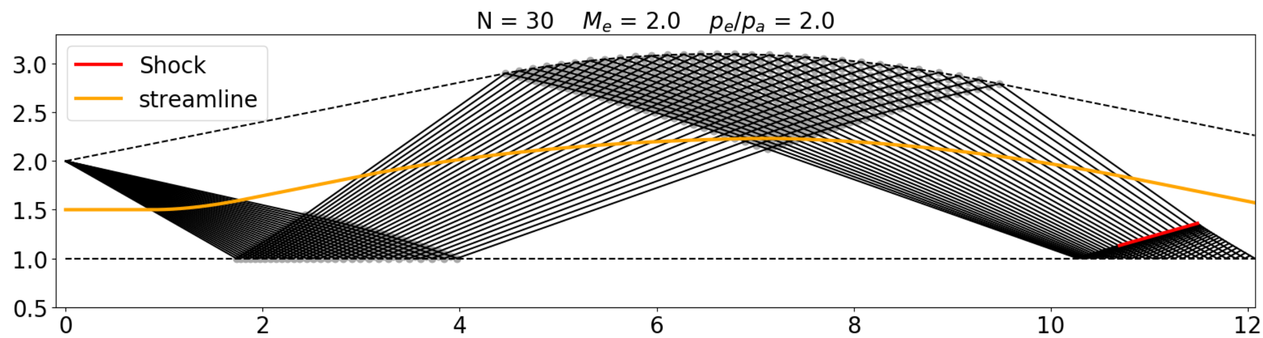

Figure 5: Streamline starting at $y = 3H/4$, $M_e = 2.0$, $p_e/p_a = 2.0$

Again, the orange line describes the streamline throughout the jet downstream. This time, the streamline varies in height and is seen to be closely following the shape of the jet boundary after the first interaction with the expansion fan. The streamline first crosses the first simple region before shortly remaining in the uniform region and proceeding to the second simple region. It then briefly goes through the non-simple region beneath the jet boundary and passes through the last simple region (that is present before the shock).

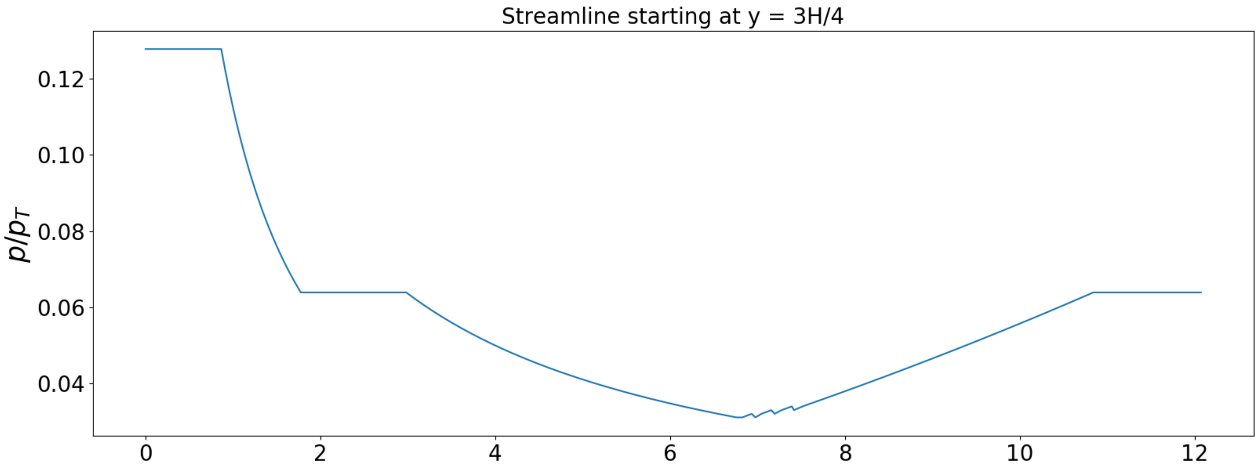

Figure 6: Pressure plot along streamline starting at $y = 3H/4$, $M_e = 2.0$, $p_e/p_a = 2.0$

The pressure plot for this streamline has more features. The pressure remains constantly initially for the uniform region until the simple region where it reduces exponentially. This is correct as going through an expansion fan results in higher Mach number and lower pressure. The streamline then goes through another uniform region where the pressure remains constant until it meets the second simple region in which it decreases exponentially again. As it meets the non-simple region, the zig-zag behaviour is seen again due to crossings of multiple characteristics in the region with an ultimate increase in pressure. It then has another quasi-linear increase as it goes through the last simple region before maintaining the final value of pressure as it meets the uniform region.

In conclusion, the pressure plots have depicted the behaviour of the streamline as I have expected where pressure decreases for simple region consisting of expansion waves, increases for simple region consisting of compression waves and remains constant for uniform region. However, through the plots, I have learnt that the first non-uniform region (around $x = 2$ to $x = 4$) is expansion waves dominant whereas the second non-uniform region (below jet boundary) is compression waves dominant.

5 Computation accuracy

The computations are exact for the uniform regions due to the absence of any characteristic intersections. Furthermore, at all characteristic intersections in the non-simple regions, the values of the parameters are exact as well. However, the location as to where these intersections occur are not exact. This is because we are trying to compute a non-linear regions that supposedly have curved characteristics using linear characteristics. This then of course affects the subsequent simple regions as well. Likewise, the invariant values of the characteristics are exact but the sizes of the regions are not entirely accurate.

With the computational method that was implemented in my script, for a low number of characteristics, the shock seems to appear at inaccurate locations, earlier in the downstream compared to larger number of characteristics, in the simple region bounded by corners FGH which is before the compression waves are reflected off the centerline. This is probably due to consecutive characteristics being alot further away than they are for larger number of characteristics, which leads to

6 Location of the shock

The change in location of the shock is studied in this section with respect to change in Mach number and exit pressure

6.1 Variance in Mach number

Firstly, the effect of Mach number is shown in the figures below.

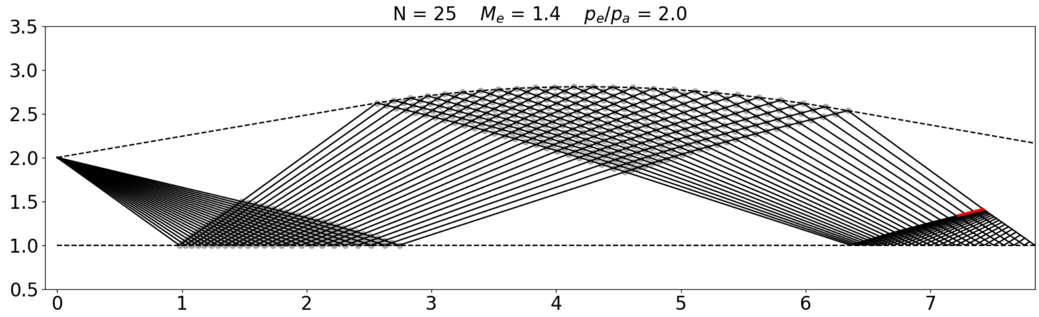

Figure 7: Characteristics pattern, $M_e = 1.4$, $p_e/p_a = 2.0$

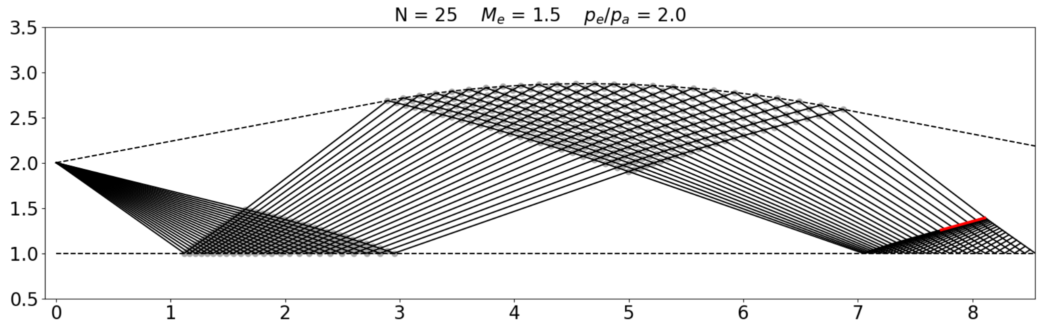

Figure 8: Characteristics pattern, $M_e = 1.5$, $p_e/p_a = 2.0$

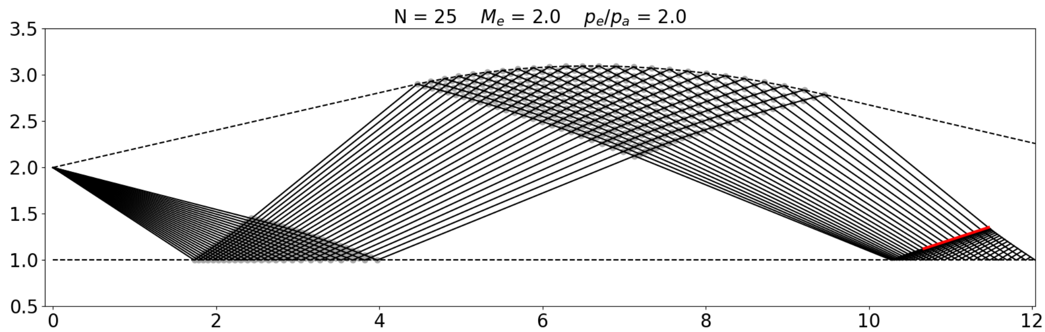

Figure 9: Characteristics pattern, $M_e = 2.0$, $p_e/p_a = 2.0$

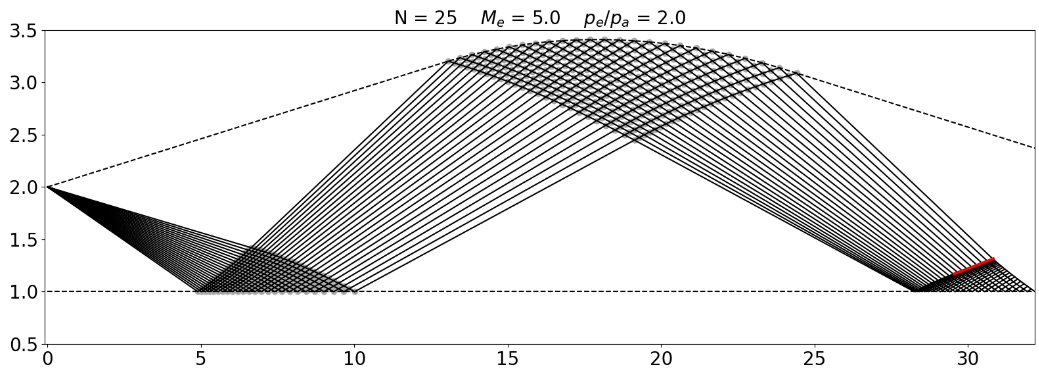

Figure 10: Characteristics pattern, $M_e = 5.0$, $p_e/p_a = 2.0$

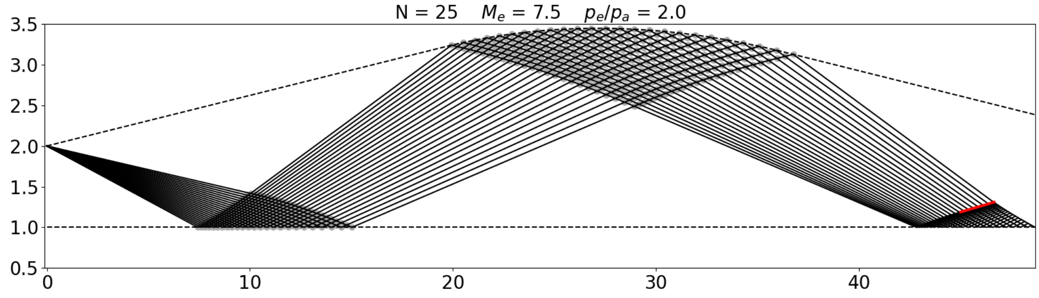

Figure 11: Characteristics pattern, $M_e = 7.5$, $p_e/p_a = 2.0$

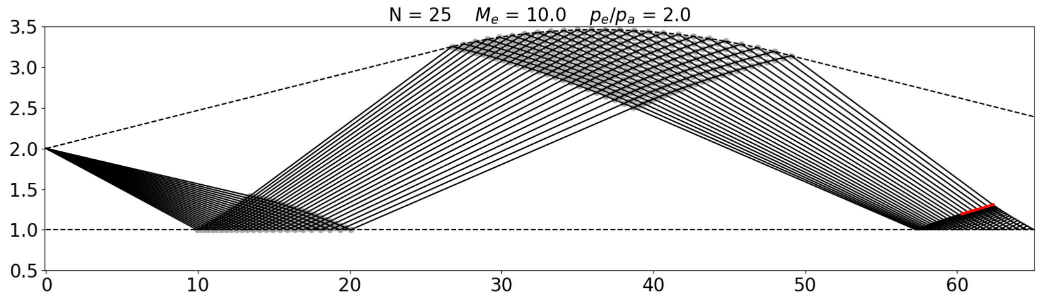

Figure 12: Characteristics pattern, $M_e = 10.0$, $p_e/p_a = 2.0$

It can be observed that compared to $M_e = 2$, the shock (indicated by the red lines) forms later for both smaller and larger values of M_e and this occurs in a gradual behaviour whereby the change in Mach number is proportional to the change in shock location.

6.2 Variance in pressure

The effect of exit pressure is shown in the figures below.

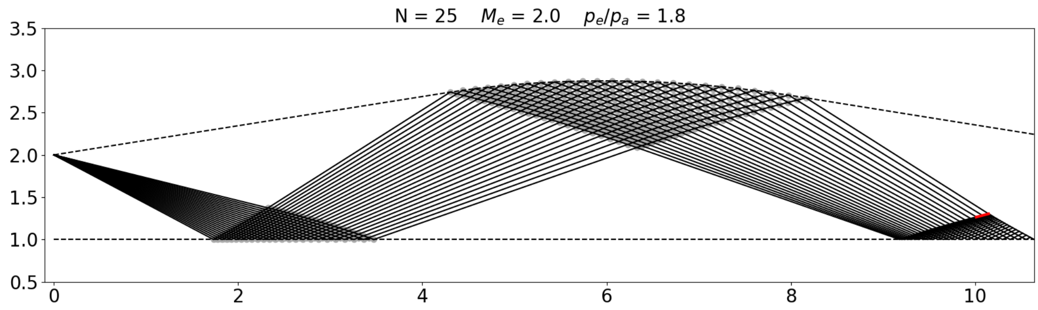

Figure 13: Characteristics pattern, $M_e = 2.0$, $p_e/p_a = 1.8$

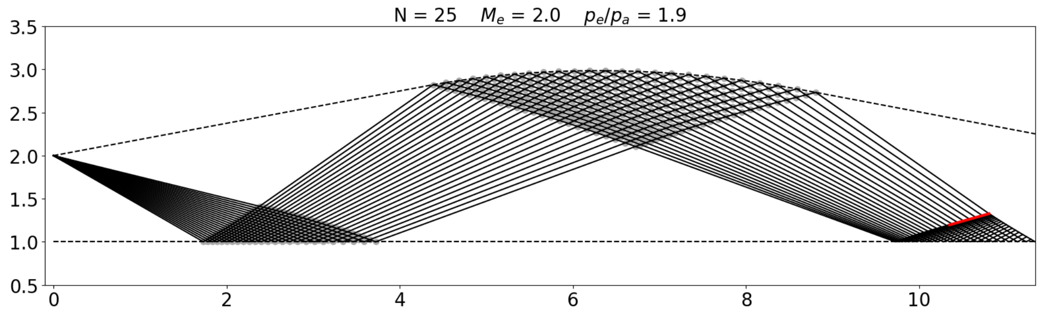

Figure 14: Characteristics pattern, $M_e = 2.0$, $p_e/p_a = 1.9$

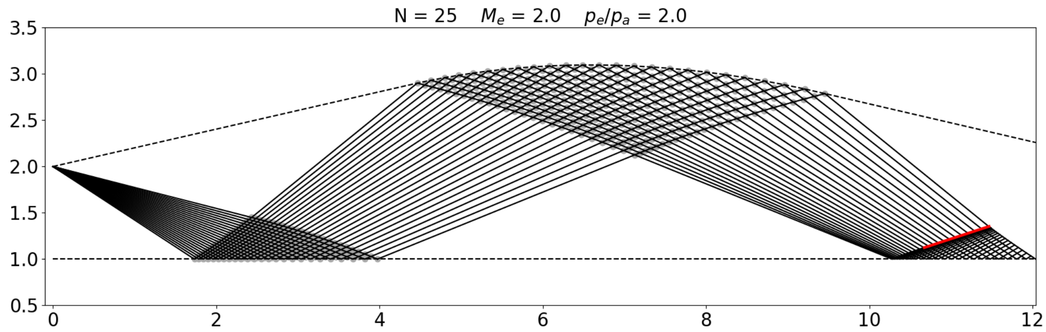

Figure 15: Characteristics pattern, $M_e = 2.0$, $p_e/p_a = 2.0$

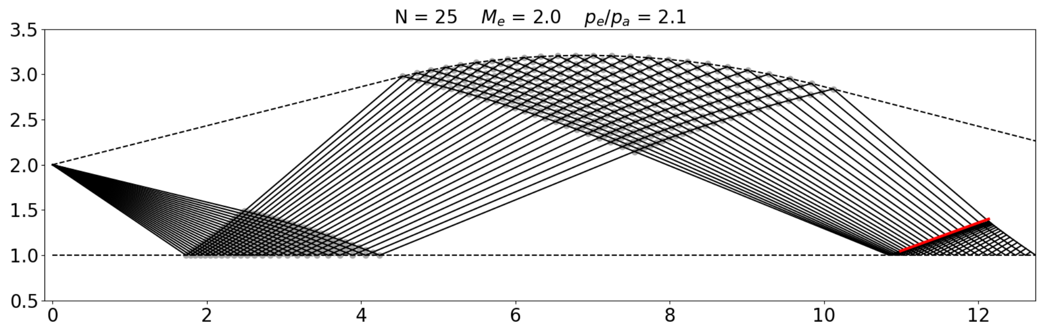

Figure 16: Characteristics pattern, $M_e = 2.0$, $p_e/p_a = 2.1$

Unlike the variance with Mach number wherein the shock location depicted a parabola pattern, appearing earlier and then delaying again, for the variance in exit pressure, the shock location shows a uni-directional behaviour. With increase in exit pressure, the shock location moves forwards, at a location nearer to the jet exit.